Hierarchical Segmentation of Hyperspectral Data

J. Anthony Gualtieri
Code 935 and Global Sciences
and Technology
NASA/GSFC
Greenbelt, MD 20771
John.A.Gualtieri.1@gsfc.nasa.gov

James C. Tilton
Code 935
NASA/GSFC
Greenbelt, MD 20771
James.C.Tilton@nasa.gov

1 Introduction

We present a new algorithm for unsupervised classification which we apply here to AVIRIS hyperspectral data. The algorithm is based on the notion of region growing in a hierarchical setting and involves both spectral clustering and spatial clustering. This is in contrast to the standard approach to performing unsupervised classification of hyperspectral data wherein pixels are clustered (segmented) into regions based only on the *similarity* of their spectra without consideration of their spatial positions. Thus an important source of information about segmentation, contiguity of spatial neighbors, is ignored. And notice that spectral clustering will yield the same clusters for a data cube, and for the same data cube where the spatial positions have been randomly permuted.

Hierarchical segmentation normally begins by assuming every pixel in the hyperspectral data cube is a separate region (however, provision is made for initialization with a per-segmentation). Then, a dissimilarity criterion is computed between spectra in neighboring regions, the minimum dissimilarity criterion is found over all pairs of neighboring regions, and all pairs of neighboring regions with this minimum dissimilarity criterion value are merged. Optionally, this spatial clustering step is followed by a spectral clustering step in which a dissimilarity criterion is computed between spectra of all spatially non-adjacent regions, and all pairs of such regions with dissimilarity less than or equal to the minimum dissimilarity value found in the spatial clustering step are merged. After sufficient iterations of this procedure are completed so that a potentially meaningful segmentation is produced, the segmentation process is checked for convergences. A convergence is signified by a jump in a global dissimilarity criterion calculated between the region mean image and the original image data values. When a convergence is detected, the image segmentation from the previous iteration is saved. The region growing process (spatial clustering plus the optional spectral clustering) is continued until there are only two regions remaining, and this two region segmentation is also saved. The outcome is a hierarchical stack of segmentations at different levels of segmentation detail in the form of a tree structure with two regions at the top which come from merges of multiple regions at the next level down, and so on, to the highest level of segmentation detail from the first detected convergence.

Clearly this process is compute intensive. However the advent of cluster computing by

Beowulf class machines [1], in which tens of commodity Intel processors running Linux are interconnected with a high speed network, allows methods requiring large computational resources to be explored. For example on a 64 processor Beowulf machine with a Myrinet network linking the processors, it is possible to perform hierarchical segmentation on a data cube of size [lines, samples, bands] = [464, 595, 182] in 16 minutes, and on a data cube of size [lines, samples, bands] = [128, 128, 182] in 129 sec.

Below we describe the hierarchical segmentations algorithm and demonstrate its use on a hyperspectral data cube taken by AVIRIS at the Patuxent Wildlife Research Center in Laurel Maryland.

2 Hierarchical Segmentation

Image segmentation is a partitioning of an image into sections or regions. These regions may be later associated with a ground cover type or land use type, but the segmentation process simply gives generic labels (region 1, region 2, etc.) to each region. The regions consist of groupings of hyperspectral image pixels that have similar data feature values. These data feature values may be the hyperspectral data values themselves, and/or they may be derived features such as band ratios [2] or textural features [3]. Image segmentation is a key first step in a number of approaches to image analysis. In image analysis, the group of pixels contained in each region provides a good statistical sampling of data values for more reliable labeling based on multi-spectral or hyperspectral feature values. In addition, the region shape can be analyzed as an additional clue for the appropriate labeling of the region. Most image segmentation approaches can be placed in one of three classes [4]: 1. Characteristic feature thresholding or clustering, 2. Boundary detection, 3. Region growing.

Characteristic feature thresholding or clustering is often ineffective because it does not exploit spatial information. Boundary detection does exploit spatial information through examining local edges found throughout the image. For simple noise-free images, detection of edges results in straightforward boundary delineation. However, edge detection on noisy, complex images often produces missing edges and extra edges producing region boundaries that do not necessarily form a set of closed connected curves that surround connected regions. We prefer region growing because it exploits spatial information and guarantees the formation of closed connected regions. However, region growing is not without its problems. With region growing, spectrally similar but spatially disjoint regions are never associated together, and it is often not clear at what point the region growing process should be terminated. Also, region growing tends to be a computationally intensive process.

Tilton has developed a hybrid region growing and spectral clustering approach (first described in [5]) that largely overcomes these problems. The hybridization with spectral clustering allows association of spectrally similar but spatially disjoint regions. The approach also includes the detection of natural convergence points to assist in determining at what point the region growing process should terminate. Finally, the recursive version of this

approach is very effectively implemented on MIMD (Multiple Instruction, Multiple Data stream) parallel computers, which greatly reduces the amount of time required to segment large images with this approach. Next we give a description of the segmentation approach. The actual implementation is on Beowulf class cluster machine at Code 935, NASA/GSFC [6].

2.1 Hybrid of Region Growing and Spectral Clustering

We begin by defining the global criterion that will control how the algorithm will decide when to merge regions of similar clusters, and how the algorithm will decide that the particular level reached in the merging process is a natural convergence point and should be saved. Then we describe a high-level outline of the hybrid image segmentation (HSEG) approach.

2.1.1 Dissimilarity Criterion

Selection of an appropriate dissimilarity criterion is generally dependent on the application the resulting segmentations will be used for, and on the characteristics of the image data. However in earlier studies, Tilton [7] found the *Euclidean Spectral Distance* to be a useful criterion.

Let two regions (which may be individual pixels or regions of contiguous or non-contiguous pixels that are spectrally similar), be labeled by ω, ψ. For these regions we can compute mean spectral vectors $\bar{\mathbf{x}}_\omega = (\bar{x}_{\omega 1}, \bar{x}_{\omega 2}, \ldots, \bar{x}_{\omega n_b})$ and $\bar{\mathbf{x}}_\psi = (\bar{x}_{\psi 1}, \bar{x}_{\psi 2}, \ldots, \bar{x}_{\psi n_b})$, where $\bar{x}_{\omega q} = \sum_{i \in \omega_q} x_{iq} / \sum_{i \in \omega_q} 1$ and similarly for $\bar{x}_{\psi q}$, and where q $(1 \leq q \leq n_b)$ is the index of the spectral band and n_b is the total number of bands per pixel.

From these a Euclidean spectral distance defines the dissimilarity criterion,

$$\mu_{\omega\psi} = \frac{\left[\sum_{q=1}^{n_b} (\bar{x}_{\omega q} - \bar{x}_{\psi q})^2\right]^{\frac{1}{2}}}{n_b}.$$

2.1.2 Global Criterion

The global criterion is used to identify significant changes in the segmentation results from one iteration to the next. This criterion is defined like the dissimilarity criterion, except that it compares the original image data with the region mean image from the current segmentation. The value of this criterion is calculated at each image point, and averaged over the entire image. Thus we define the global criterion as,

$$\mu_{glbl} = \frac{\sum_{\omega \in \{\alpha, \beta, \ldots\}} \sum_{i \in \omega} \left[\sum_{q=1}^{n_b} (x_{\omega q} - x_{iq})^2\right]^{\frac{1}{2}}}{n_s n_l n_b}$$

where i indexes an image pixel, ω indexes over regions of spectrally similar pixels given by the current segmentation $\{\alpha, \beta, \ldots\}$, q indexes over the all the spectral channels $1 \ldots n_b$ for each pixel or region, and n_s and n_l are the number of samples (columns) and lines (rows) in the image.

Next we give in pseudo code the Hierarchical Segmentation Algorithm. This is the basic algorithm, but in order to render it computationally on any but small hyperspectral cubes it will need to be embedded in a recursive hierarchical segmentation algorithm which is given in the sequel. We will show two ways it can be used depending on the value of the parameter N_{chk}:

$$N_{chk} \begin{cases} = 0 & \text{suppresses convergence checking} \\ > 0 & \text{uses convergence checking} \end{cases}$$

2.1.3 The Hierarchical Segmentation Algorithm

seg = **HSEG(** $cube, N_{min}, N_{chk}, \phi_{thresh}, seg^{init}$)

% $cube$ hyperspectral cube of size r, s, n_b
% N_{min} number of regions threshold. When called by **recur_hseg** with $n_{rec} > 0$, set $\gg 2$, else 2
% N_{chk} number of regions threshold when level 0 has converged.
% ϕ_{thresh} threshold for natural global convegence test
% seg^{init} initial hyperspectral cube segmentation
%Define: k the heirarchical level index
% $N^{(k)}$ number of regions in level k
% τ_{loop} control output

$k = 0,\ N^{(0)} = r \times s,\ seg = seg^{init},\ \mu_{glbl}^{(pre)} = 0,\ \tau_{loop}$ =**true** % initialization

while($N^{(k)} > N_{min}$) % outer loop with end criterion
while(τ_{loop} == **true** & $N_{chk} > N_{min}$) % inner loop with natural convergence criterion
compute $\mu_{\omega\psi}$ between all pairs of % region growing
spatially adjacent regions, $\omega\psi$,
find smallest $\mu_{adj}^{(min)}$ over all spatially % region growing
adjacent regions
$seg \leftarrow$**merge(**seg, all pairs of spatially adjacent regions
ϵ and ν where $\mu_{\epsilon\nu} = \mu_{adj}^{(min)}$) % region growing
$N^{(k)} \leftarrow$ new number of regions
compute $\mu_{\sigma\tau}$ where σ and τ index over % spectral clustering
all pairs of non-spatially adjacent regions.
$seg \leftarrow$**merge(** seg, all pairs of non-spatially adjacent
regions λ and ξ where $\mu_{\lambda\xi} \leq \mu_{adj}^{(min)}$) % spectral clustering
$N^{(k)} \leftarrow$ new number of regions
if ($N^{(k)} \leq N_{chk}$) % natural convergence test begins at N_{chk}
compute $\mu_{glbl}^{(curr)}$.
if ($\mu_{glbl}^{(pre)} > 0$) $\phi = \mu_{glbl}^{(curr)} / \mu_{glbl}^{(pre)}$ % compute ϕ only when defined
else $\phi = \phi_{thresh}$
$\mu_{glbl}^{(pre)} = \mu_{glbl}^{(curr)}$
if($\phi \geq \phi_{thresh}$) $\tau_{loop} \leftarrow$ **false**
end if
end while
if (τ_{loop} ==**false**)
save region label map from the iteration k
as the segmentation result for level k.
$\tau_{loop} \leftarrow$**true**
$k \leftarrow k + 1$
endif
end while
if($N_{chk} > 0$)
save region label map from the current iteration as the
coarsest instance of the final hierarchical segmentation result.
stop
end if
end

2.1.4 Implementation Overview

A practical implementation of the hierarchical segmentation algorithm for all but the smallest hyperspectral cubes requires that the combinatorial growth in inter-region comparisons in the spectral clustering steps be addressed. Tilton's solution, is to recursively subdivide the image data into smaller and smaller sections until an image size is reached in which the required number of inter-region comparisons is sufficiently constrained that the **HSEG** algorithm can be applied without the global criterion, and run until a pre chosen number of regions, N_{min} has been found. At this point the segmentation so found is passed to the **HSEG** algorithm *with* the global criterion, which then produces the stack of region labeled segmentations. The number in the stack is dependent on the global criterion ϕ_{thresh} and the number of regions that initializes the **HSEG** computation [8]. Additional processing speed can be obtained through a parallel implementation of the recursive hierarchical segmentation algorithm [9]. However, the recursive decomposition and subsequent recombination of results can easily impart processing window artifacts in the segmentation results so obtained unless steps are taken to remove the artifacts. Effective and efficient methods for removing these artifacts have been devised [10]. This is called the **RHSEG** algorithm and is given in the sequel.

2.1.5 Implementation Details

For recursion, the original hyperspectral cube is subdivided along the spatial coordinates into 4 equal sized cubes rectangular in the spatial coordinates, each of which is itself subdivided into 4 equal rectangles and so on n_{rec} times until each of the $4^{n_{rec}}$ cubes resulting are small enough to directly apply **HSEG** to. Typically we choose n_{rec} such that the number of pixels in the smallest cube, $n_l/2^{n_{rec}} \times n_s/2^{n_{rec}}$, is in the range 500...2000. In order for the number of pixels in the two spatial directions be integers at level n_{rec} we pad out the original cube to $n_l^{(pad)}$ and $n_s^{(pad)}$ with spectra of all 0's so that $n_l^{(pad)}/2^{n_{rec}}$ and $n_s^{(pad)}/2^{n_{rec}}$ are both integers. The recursive algorithm, called **RHSEG**, first establishes the number of recursions and then calls recursively calls **recur_hseg** until the resulting subdivided hyperspectral cubes can be processed, without the global criterion, by **HSEG**. The resulting subdivided segmentation are then reassembled in **recur_hseg** until a segmentation at the top level of the recursion is obtained. This then initializes a final call of **HSEG** with the global criterion, and produces the output stack of region labeled segmentations.

$seg(n_l, n_s)$ = **RHSEG** $(cube(n_l, n_s, n_b), N_{min}, N_{chk}, \phi_{thresh}, seg^{init})$
% $(cube(n_l, n_s, n_b)$ initial data cube
% N_{min} number of regions threshold.
% N_{chk} number of regions threshold when level 0 has converged.
% ϕ_{thresh} threshold for natural global convegence test
% seg^{init} optional initial hyperspectral cube segmentation
% Define: n_{rec} number of recursive calls
% l_{rec} recursion level
% $n_l^{(pad)}$, $n_s^{(pad)}$ padded sizes

compute n_{rec} as an integer such that $\frac{1}{2}\log_2[n_l n_s/2000] \leq n_{rec} \leq \frac{1}{2}\log_2[n_l n_s/500]$
$n_l^{(pad)} = 2^{n_{rec}}\lceil n_l/2^{n_{rec}}\rceil$
$n_s^{(pad)} = 2^{n_{rec}}\lceil n_s/2^{n_{rec}}\rceil$
$cube(n_l^{(pad)}, n_s^{(pad)}, n_b)$ = **pad(** $cube(n_l, n_s, n_b)$ **)** % pad out with 0 spectra
$l_{rec} = 0$
seg = **recur_hseg(** $cube(n_l^{(pad)}, n_s^{(pad)}, n_b), N_{min}, N_{chk}, \phi_{thresh}, n_{rec}, l_{rec}, seg^{init}$ **)**
end

seg = **recur_hseg(** $cube(r, s, n_b), N_{min}, N_{chk}, \phi_{thresh}, n_{rec}, l_{rec}, seg^{init}$ **)**
% $cube(r, s, n_b)$ sub-cube of size $r \times s \times n_b$
% N_{min} minimum number of regions for stopping criterionn
% N_{chk} number of regions threshold when level 0 has converged.
% ϕ_{thresh} threshold for natural global convegence test
% n_{rec} number of recursive calls desired
% l_{rec} recursion level
% seg^{init} optional initial hyperspectral cube segmentation
%Define i index $\in 1, 2, 3, 4$ naming i'th sub-cube in recursion
if($l_{rec} < n_{rec}$ **true)**
for i=1,4 **do**
$cube_i(r/2, s/2, n_b)$ = **split4(**$i, cube(r, s, n_b)$ **)**
$seg_i^{init}(r/2, s/2, n_b)$ = **split4(**$i, seg^{init}(r, s, n_b)$ **)**
$seg_i(r/2, s/2, n_b)$ = **recur_hseg(** $cube_i(r/2, s/2, n_b), N_{min}, N_{chk}, \phi_{thresh},$
$n_{rec}, l_{rec} + 1, seg_i^{init}(r/2, s/2, n_b$ **)** % Incremented l_{rec}
end do
$seg^{init}(r, s, n_b)$ = **concatenate(** $seg_1, seg_2, seg_3, seg_4$ **)**
end if
if($l_{rec} == 0$ & $N_{chk} > N_{min}$ **)** $N_{min} = N_{chk}$
seg^{temp} = **HSEG(** $cube, N_{min}, 0, \phi_{thresh}, seg^{init}$**)** % Note, set $N_{chk} = 0$
seg^{init} = **eliminate_artifacts(** seg^{temp} **)**
if($l_{rec} == 0$ **)**
$N_{min} = 2$
seg = **HSEG(** $cube, N_{min}, N_{chk}, \phi_{thresh}, seg^{init}$**)**
end if
return
end

As described above, the **RHSEG** algorithm is prone to producing processing window artifacts in its segmentation results. These processing window artifacts are eliminated by examining the segmentation results after the completion of the call to **HSEG** in **recur_hseg**, and changing the region membership of pixels that are more similar to a region other than

the region the pixels are currently assigned to. How this process is efficiently implemented is described in [10].

2.1.6 Parameter Settings and Program Refinements

As described in sections 2.1.3 and 2.1.4 above, the **HSEG** and **RHSEG** algorithms with the Euclidean spectral distance dissimilarity criterion tend to produce segmentations with a large number of small regions. Since these small regions are generally of little use in the ultimate analysis of the segmentation results, an option has been added to the **HSEG** and **RHSEG** algorithms to encourage these small regions to merge into other regions. When this option is selected, the dissimilarity function is multiplied by the factor

$$1 - \left[\frac{P_m - Ps}{P_m}\right],$$

where P_m is a user supplied parameter, and P_s is the number of pixels in the smaller of the two regions being compared. This bias factor is applied only when $P_s < P_m$. In this paper we have used $P_m = 16$, $N_{min} = 256$, $N_{chk} = 64$, and $\phi = 1.01$.

3 Results for Patuxent Wildlife Research Center

On August 1, 2001 a series of AVIRIS scenes along a single flight line which included the Patuxent Wildlife Research Center (PWRC) in Laurel, Maryland was acquired. PWRC is a research facility of the U.S Fish and Wildlife Services and is roughly 50 square km and comprises mostly forest with some meadows, lakes, ponds, wetlands, and the Patuxent River. The data was radio-metrically corrected and geo-registered with a pixel size of 12.1 m. Scene 3 of flight-line, f010801t01p03_r08, was corrected for cross track illumination variation, due to the flight-line being flown at +60 deg from true north, and the bands were subset from 224 to 182 by removing bands with negative values due to water absorption and noise. To demonstrate the algorithm, a square subset of size 128 $\times$ 128 was subset out [11]. A preliminary use of recursive hierarchical segmentation **RHSEG** was performed to create masks for the clouds and cloud shadows in the scene. In this case 17 levels were found as shown in Fig. 1.

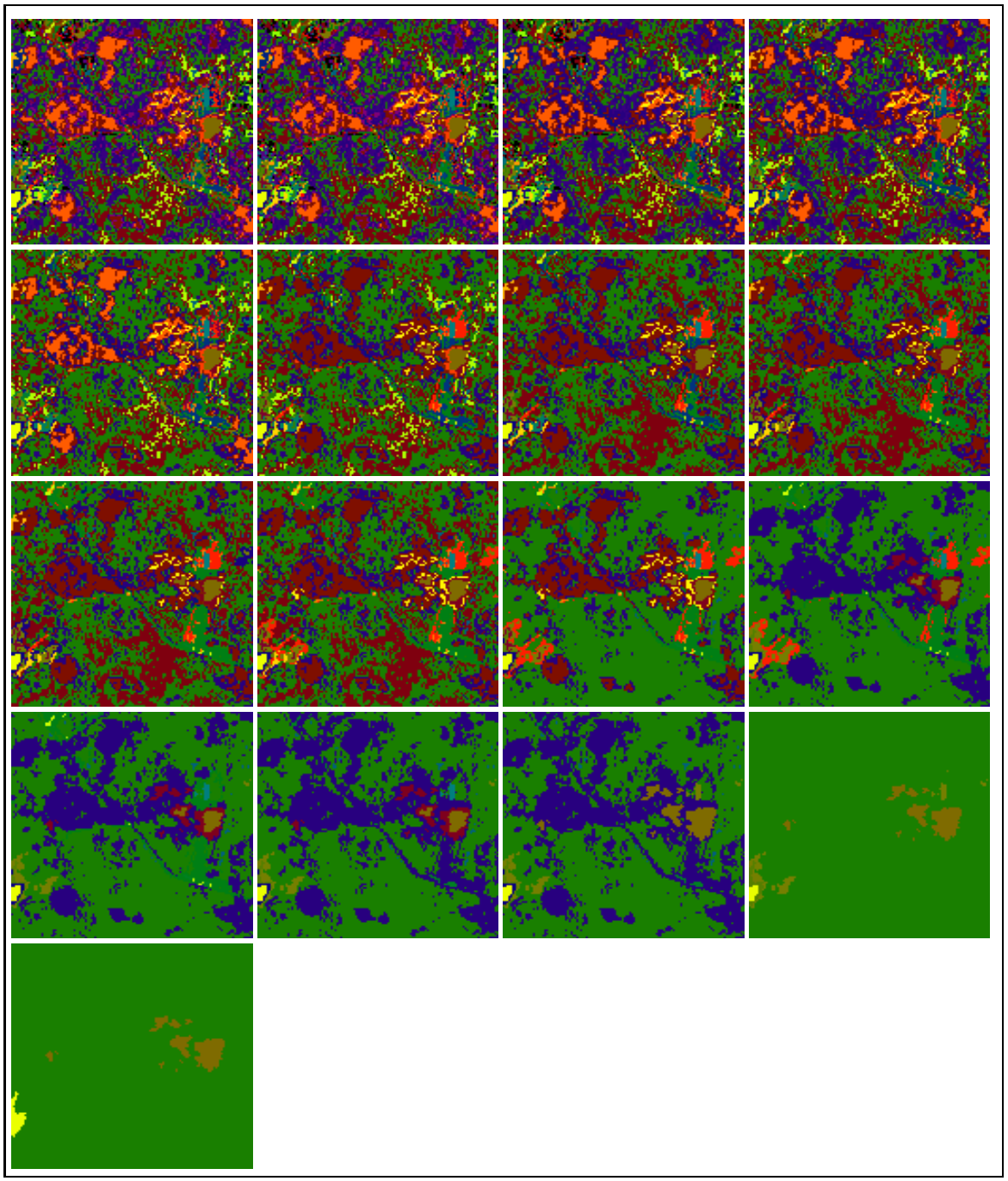

Fig. 1: The 17 levels of the hierarchical segmentation with an arbitrary color map that is the same for each level. Level 1 is the top left and follows left to right and then down. Note that there are 64 different colors in level 1, but they cannot all be resolved. At level 17 there are only 3.

This involves the use of the region labeling tool developed by Tilton [12]. The tool provides a Gui, typically with two windows, one showing a reference image such as an approximate true color image, and a second window in which the user can move up and down the segmentation hierarchy. The mouse is used as a pointing tool to track and show

the same spatial location in both windows, and the mouse buttons allow the user to select a single region at any level in the hierarchy window and then track that region up and down, as it either remains the same or coalesces with other regions going up or remains the same or splits into sub-regions going down the hierarchy.

Thus a feature in the reference image is used to select regions in the hierarchy. If, in tracking that region of the hierarchy, it is found to be stable over several levels, then it is taken to mean it is a region in the final segmentation. The user then can select a color from a palette, and a label for the region. For example in Fig. 1 at level 16 the yellow, brown, and tan regions match the clouds and cloud shadows, and these regions persist, though resolving to additional colors with the same external boundary down to level 10, The region labeling tools allows the selection of those three regions at level 17 to then be tracked down to lower levels to see how those regions break up into sub regions. To the extent that they maintain the same shape over several levels is an indication of their describing a real region feature in the scene. The analyst can also use the region labeling tool to select regions of interest by enclosing them with drawn closed loops in order to add pixels to a region in an existing segmentation.

In this way a cloud/cloud shadow mask was derived. Then the mask was applied to the original hyperspectral cube and **RHSEG** was applied to the masked hyperspectral cube. That resulted in the segmentation shown in Fig. 2, for which 18 levels occurred.

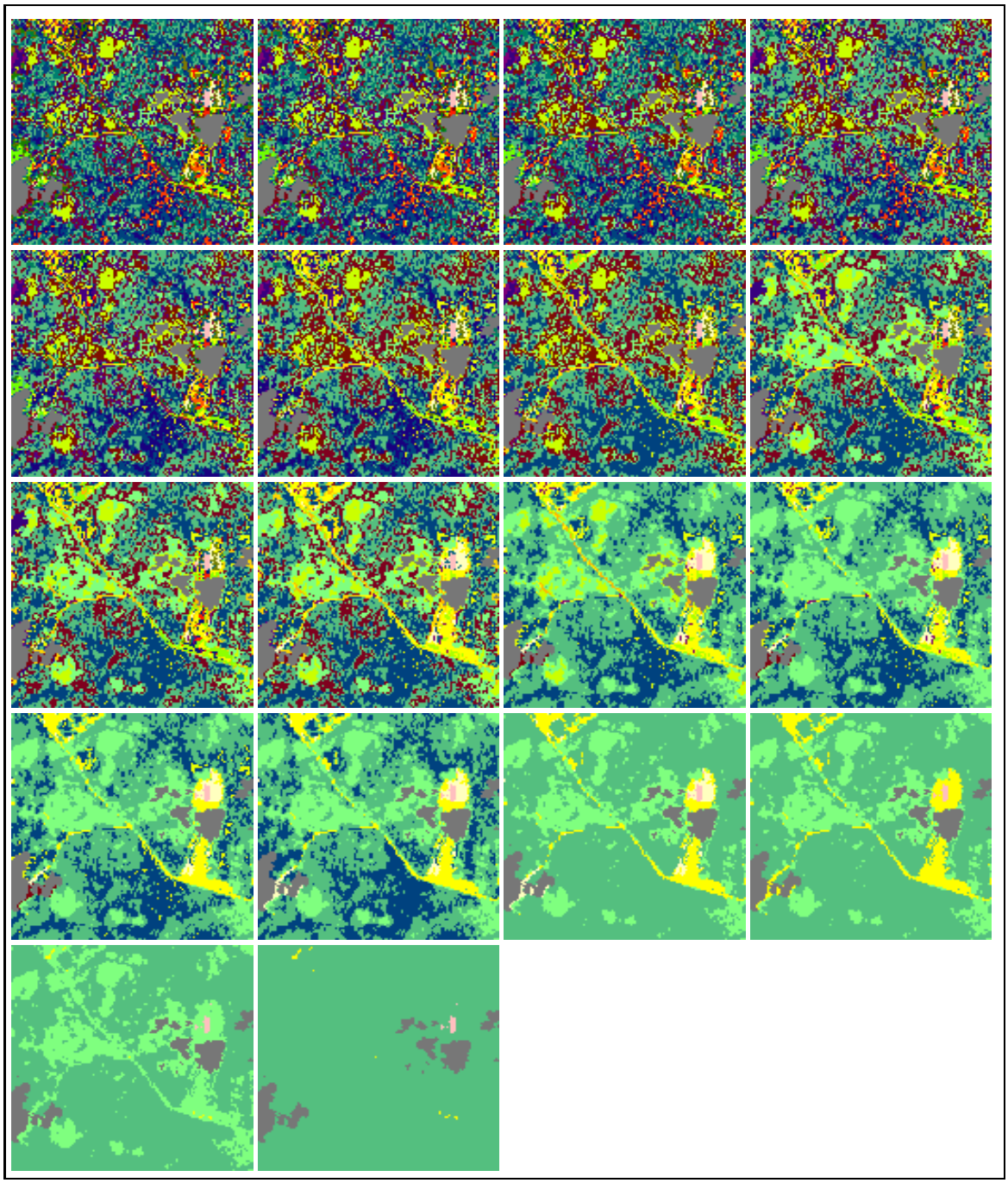

Fig. 2: The 18 levels of the hierarchical segmentation with a cloud mask, in grey, where the arbitrary color map as used in Fig. 1 has been changed so that the colors agree with color choices used in the right part of Fig. 3. This was done after the user manually selected the colors for the classification in the right part of Fig. 3 using the region labeling tool. Then the default color map was modified to substitute these colors for the original colors in this segmentation to make it easier to see the relationship of this segmentation to the result in the right of Fig. 3.

A second application of the region labeling tool to the segmentation in Fig. 2 was then used to obtain the result shown in the right of Fig. 3. The approximate true color image is shown in the left of Fig. 3. Note that segmentation result in the right of Fig. 3. is close, but not identical to level 15 in Fig. 2. For example the road features (in yellow) are continuous in our final segmentation, are broken in level 15, but are seen to be continuous in levels 11 down to 6 in Fig. 2.

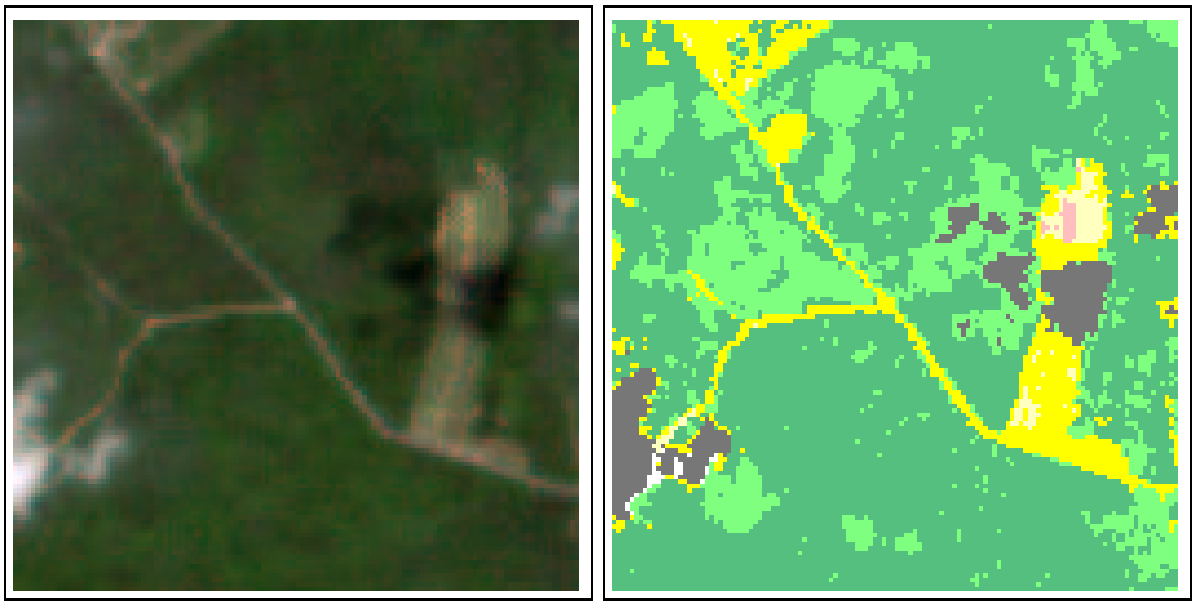

Fig. 3: On the left the approximate true color image, and on the right a classification based on cloud/cloud shadow masking followed by hierarchical segmentation and subsequent use of the region labeling tool. Here gray represents the cloud/cloud shadow mask. Note it is similar, but not identical to level 15 in Fig. 2

4 Conclusion

The hierarchical segmentation algorithm has been described together with its implementation on a Beowulf computer using a recursive approach. The algorithm performs both spatial region growing and spectral clustering, thereby using both spectral and spatial information in the hyperspectral cube. It yields a new representation of hyperspectral imagery as a hierarchy of images that show how regions coalesce as one ascends the hierarchy. By examining the stability of regions across levels in the hierarchy, a segmentation can be found that can form the basis for unsupervised classification. To help in this examination across levels, the region labeling tool is introduced to assist an analyst in extracting these stable features. We have demonstrated the algorithm and the region labeling tool on a small AVIRIS hyperspectral cube, and obtained a reasonable segmentation into recognizable features. While this is a preliminary result, and it does involve the analyst interpreting the scene, the hierarchical segmentation together with the region labeling tool provides a new method to analyze hyperspectral data. In the future we plan to further automate this process by taking advantage

of the rich representation of the data that the hierarchical segmentation provides.

Acknowledgment

J. A. Gualtieri would like to thank Prof. Shunlin Liang of the Department of Geography at the University of Maryland, College Park for adding the low altitude flight line over PWRC used in this work to his AVIRIS flight request in 2001.

References

[1] T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*, volume I, Architecture, pages I:11–14, Boca Raton, FL, August 1995. CRC Press.

[2] R. A. Schowengerdt. *Remote Sensing, Models and Methods for Image Processing*, chapter 5. Academic Press, Chestnut Hill, MA, 2nd ed. edition, 1997.

[3] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67:786–804, 1979.

[4] K. S. Fu and J. K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[5] J. C. Tilton. Image segmentation by region growing and spectral clustering with a natural convergence criterion. In *Proc. of the 1998 International Geoscience and Remote Sensing Symposium*, July 1998.

[6] This is a Beowulf class cluster machine consisting of 66 nodes each a Pentium PRO PC with 2 CPU's where 64 nodes are for computation and 2 are for control. See `http://newton.gsfc.nasa.gov/thehive`.

[7] J. C. Tilton. A recursive pvm implementation of an image segmentation algorithm with performance results comparing hive and the cray t3e. In *Proceeding of the 7th Symposium on Massively Parallel Computation*, pages 146–153, Feb. 1999.

[8] J. C. Tilton. Method for recursive hierarchical segmentation by region growing and spectral clustering with a natural convergence criterion, Feb 2000. Disclosure of Invention and New Technology (Including Software): NASA Case No. GSC 14,328-1, NASA's Goddard Space Flight Center.

[9] J. C. Tilton. Method for implementation of recursive hierarchical segmentation on parallel computers, Feb. 2000. Disclosure of Invention and New Technology (Including Software): NASA Case No. GSC 14,305-1, NASA's Goddard Space Flight Center.

[10] J. C. Tilton. A method for recursive hierarchical segmentation which eliminates processing window artifacts, Oct. 2002. Disclosure of Invention and New Technology (Including Software): NASA Case No. GSC 14,681-1, NASA's Goddar Space Flight Center.

[11] The subset was obtained by first subsetting [lines, samples] = [650:1113, 54:648] from scenes 2 of AVIRIS flightline f010801t01p03_r08 taking [1,1] as the upper left hand cornet. From this a subset of size 128 by 128 was taken. The original 224 bands were subsetted to 182 by removing bands [4, 77, 107-115, 153-169, 209, 211-212, 214-224] which were found to have negative values, indicating noise or water absorption. An approximate true color image can be obtained by selecting bands [r, g, b] = [34, 19, 11] from the 182 band data cube.

[12] J. C. Tilton. A region labeling tool for use with hierarchical segmentation, Feb. 2000. Disclosure of Invention and New Technology (Including Software): NASA Case No. GSC 14,331-1, NASA's Goddard Space Flight Center, February 29, 2000.